

UNCLASSIFIED

AD . 4 2 6 4 5 5

DEFENSE DOCUMENTATION CENTER

FOR

SCIENTIFIC AND TECHNICAL INFORMATION

CAMERON STATION, ALEXANDRIA, VIRGINIA



UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

426453

MEMORANDUM
RM-3815-PR
DECEMBER 1963

SEARCHED INDEXED SERIALIZED FILED
AS

DYNAMIC PROGRAMMING AND ILL-CONDITIONED LINEAR SYSTEMS

Richard Bellman, Robert Kalaba and Joanne Lockett

PREPARED FOR:

UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM

RM-3815-PR

DECEMBER 1963

**DYNAMIC PROGRAMMING AND
ILL-CONDITIONED LINEAR SYSTEMS**

Richard Bellman, Robert Kalaba and Joanne Lockett

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

DDC AVAILABILITY NOTICE

Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC).

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

As part of its Project RAND research program, RAND engages in basic support studies in mathematics. The present Memorandum is concerned with computer techniques combining several methods of approximation.

--v--

SUMMARY

Dynamic programming, successive approximations, extrapolation, and smoothing are used in this RAND Memorandum to treat ill-conditioned systems. Numerical examples are given.

CONTENTS

PREFACE	iii
SUMMARY	v
Section	
1. INTRODUCTION	1
2. DYNAMIC PROGRAMMING APPROACH	2
3. EXAMPLE	5
4. SUCCESSIVE APPROXIMATIONS	6
5. APPLICATION OF SUCCESSIVE APPROXIMATION PLUS SMOOTHING	10
6. EXTRAPOLATION	12
7. COMBINATION OF METHODS	14
8. DISCUSSION	14
Appendix	
COMPUTING ROUTINE	19
REFERENCES	25

DYNAMIC PROGRAMMING AND ILL-CONDITIONED LINEAR SYSTEMS

1. INTRODUCTION

A system of linear equations

$$(1.1) \quad Ax = b$$

is said to be ill-conditioned if A is almost singular. The computational solution will, in general, require some additional devices since the numerically large elements in A^{-1} amplify small changes in b , and small numerical errors incurred in the course of the calculation, to the point that they overwhelm the significant elements.

In a number of situations, particularly in those of engineering and physical origin, we have some a priori information concerning the solution x . For example, we may know that $\|x - c\|$ is small, where c is a known vector, or we may know that the components of x are monotone increasing,

$$(1.2) \quad x_1 \leq x_2 \leq \cdots \leq x_N.$$

In the first case, we can use this additional information by considering the new problem of minimizing the quadratic form

$$(1.3) \quad (Ax - b, Ax - b) + \lambda(x - c, x - c),$$

where (x, y) denotes the usual inner product and λ is a parameter to be chosen adroitly. This technique has

been used by Twomey [1] and Phillips [2].

In the second case, we could consider the problem of minimizing the quadratic form

$$(1.4) \quad (\mathbf{Ax} - \mathbf{b}, \mathbf{Ax} - \mathbf{b}) + \lambda(\nabla^2 \mathbf{x}, \nabla^2 \mathbf{x}),$$

where $\nabla^2 \mathbf{x}$ represents the vector whose i -th component is $x_i - 2x_{i-1} + x_{i-2}$, with $x_0 = x_{-1} = 0$.

In this paper we shall confine our attention to a consideration of the problem posed in (1.3). There are several features of novelty to our treatment. In the first place, we shall employ a dynamic programming algorithm, cf. [3], [4], to determine the minimum of the expression in (1.3). Secondly, we shall combine this with successive approximations in \mathbf{x} and extrapolation in λ . Examples will be given of the efficacy of these methods.

2. DYNAMIC PROGRAMMING APPROACH

To apply dynamic programming to the minimization of the quadratic form

$$(2.1) \quad R_N(\mathbf{x}) = (\mathbf{Ax} - \mathbf{b}, \mathbf{Ax} - \mathbf{b}) + \lambda(\mathbf{x} - \mathbf{c}, \mathbf{x} - \mathbf{c}),$$

we observe that the minimum value is a function of \mathbf{b} , and indeed a quadratic function of \mathbf{b} .

Consider the more general problem of minimizing

$$(2.2) \quad R_M(x) = \left[\lambda(x_1 - c_1)^2 + \lambda(x_2 - c_2)^2 + \cdots + \lambda(x_M - c_M)^2 + \sum_{i=1}^N \left(\sum_{j=1}^M a_{ij}x_j - b_i \right)^2 \right],$$

where M may be any integer between 1 and N .

If $M = N$, we have the original problem. For $M = 1$, we have the simple problem of minimizing

$$(2.3) \quad \lambda(x_1 - c_1)^2 + \sum_{i=1}^N (a_{i1}x_1 - b_i)^2.$$

Let $a^{(M)}$ be the column vector

$$(2.4) \quad a^{(M)} = \begin{pmatrix} a_{1M} \\ a_{2M} \\ \vdots \\ a_{NM} \end{pmatrix}.$$

The principle of optimality yields the recurrence relation

$$(2.5) \quad f_M(b) = \min_{x_M} [\lambda(x_M - c_M)^2 + f_{M-1}(b - x_M a^{(M)})]$$

for $M \geq 2$ (cf. [3], [4]), with $f_1(b)$ defined by (2.3).

In order to use this recurrence relation, we use the fact mentioned above that $f_M(b)$ is a quadratic function of b ,

$$(2.6) \quad f_M(b) = (b, Q_M b) + 2(p_M, b) + r_M,$$

where Q_M is an $N \times N$ matrix, p_M is an N -dimensional vector, and a_M is a scalar.

Using (2.5), some direct algebraic calculations which we need not reproduce here yield the recurrence relations

$$(2.7) \quad Q_M = Q_{M-1} - \frac{A^{(M)}}{\lambda + K_M},$$
$$P_M = P_{M-1} - \frac{(\lambda c_M + p_M)\alpha^{(M)}}{(\lambda + K_M)},$$
$$r_M = r_{M-1} + \lambda c_M^2 - \frac{(\lambda c_M + p_M)^2}{(\lambda + K_M)},$$

where the auxiliary quantities in (2.7) are defined by

$$(2.8) \quad \alpha^{(M)} = Q_{M-1} a^{(M)},$$
$$p_M = (p_{M-1} a^{(M)}),$$
$$K_M = (\alpha^{(M)}, a^{(M)}),$$

and

$$(2.9) \quad A^{(M)} = \alpha^{(M)} \otimes \alpha^{(M)};$$

here \otimes denotes the Kronecker product,

$$(2.10) \quad x \otimes y = (x_i y_j), \quad i, j = 1, 2, \dots, N.$$

If we take

$$(2.11) \quad Q_0 = I, \quad p_0 = 0, \quad r_0 = 0,$$

we can use the recurrence relation of (2.7) to obtain all Q_M , p_M , r_M for $M \geq 1$. This saves computing time. The minimizing value of x_M is

$$(2.12) \quad x_M = \frac{\lambda c_M + (p_{M-1}, a^{(M)}) + (a^{(M)}, Q_{M-1} b)}{\lambda + (a^{(M)}, Q_{M-1} a^{(M)})}$$

A count of storage requirements and an estimate of times shows that the foregoing is a simple feasible procedure for systems of dimension up to 90, at least with a computer such as the IBM-7090. Using various devices, the dimension could be raised considerably—at some cost in time. In the examples that follow, we restrict ourselves, for illustrative purposes, to systems of dimension 11.

3. EXAMPLE

To illustrate the method in practice, consider the integral equation

$$(3.1) \quad \int_0^1 (x - y)^2 u(y) dy = \frac{x^2}{2} - \frac{2x}{3} + \frac{1}{4},$$

where the right-hand side was chosen so that the solution is $u(y) = y$.

Let us use a quadratic formula, say Simpson's rule, with 11 equally spaced points, $y = 0, 0.1, 0.2, \dots, 1$. Letting x assume these same values, we obtain a system of linear equations

$$(3.2) \quad \sum_{j=1}^{11} a_{ij}x_j = b_i, \quad i = 1, 2, \dots, 11,$$

where $A = (a_{ij})$ and $b = (b_1, b_2, \dots, b_{11})$ are known.

In Table 1, we show the results of various choices of c and λ contrasted with the solution attempted by direct methods and the exact solution.

It is clear that some additional effort is required to obtain a good approximation to the solution.

Figure 1 shows the instability of the solution obtained by inversion using values of $g(x) = x^2/2 - 2x/3 + 1/4$ accurate to eight significant figures. The solutions obtained by means of dynamic programming are shown in Figure 2 for various values of λ using $g(x)$ accurate to only three significant figures.

4. SUCCESSIVE APPROXIMATIONS

To improve the foregoing results, we turn first to the method of successive approximations. Consider the choice of c as a first approximation and the x obtained in this way as a second approximation. Generally, having obtained x_{N-1} , let x_N be determined by the condition that it minimize

$$(4.1) \quad (Ax - b, Ax - b) + \lambda(x - x_{N-0}x - x_{N-1}).$$

It is easy to see that the process converges to a solution of $Ax = b$ for any initial choice of c and any $\lambda > 0$, provided that A is nonsingular.

TABLE 1

-7-

c	x	True	Inv.	$\lambda = 0$	0.01	0.05	0.1	0.15	0.2	0.25
0.0	0.0	0.0	- 3.501	- 5.2797	0.039287	0.029376	0.021046	0.016294	0.013271	0.011188
0.05	0.1	0.1	4.6244	- 0.54397	0.16775	0.13702	0.11247	0.098419	0.089465	0.083286
0.1	0.2	0.2	-24.994	9.1234	0.14925	0.13476	0.12476	0.11914	0.11558	0.11313
0.15	0.3	0.3	8.979	-12.361	0.24924	0.21498	0.19535	0.18476	0.17816	0.17366
0.2	0.4	0.4	- 4.2511	-24.517	0.26024	0.23671	0.22498	0.21893	0.21524	0.21275
0.25	0.5	0.5	13.876	64.633	0.41172	0.34482	0.31333	0.29758	0.28811	0.28178
0.3	0.6	0.6	- 9.72	-71.499	0.41155	0.36460	0.34274	0.33196	0.32553	0.32125
0.35	0.7	0.7	- 4.5892	-36.844	0.65464	0.52654	0.46639	0.43687	0.41931	0.40765
0.4	0.8	0.8	- 9.3717	79.199	0.60317	0.51843	0.47804	0.45822	0.44644	0.43862
0.45	0.9	0.9	2.7077	- 7.7244	0.97820	0.76015	0.65454	0.60263	0.57175	0.55127
0.5	1.0	1.0	12.782	1.7999	0.66755	0.59910	0.56544	0.54886	0.53899	0.53243

-8-

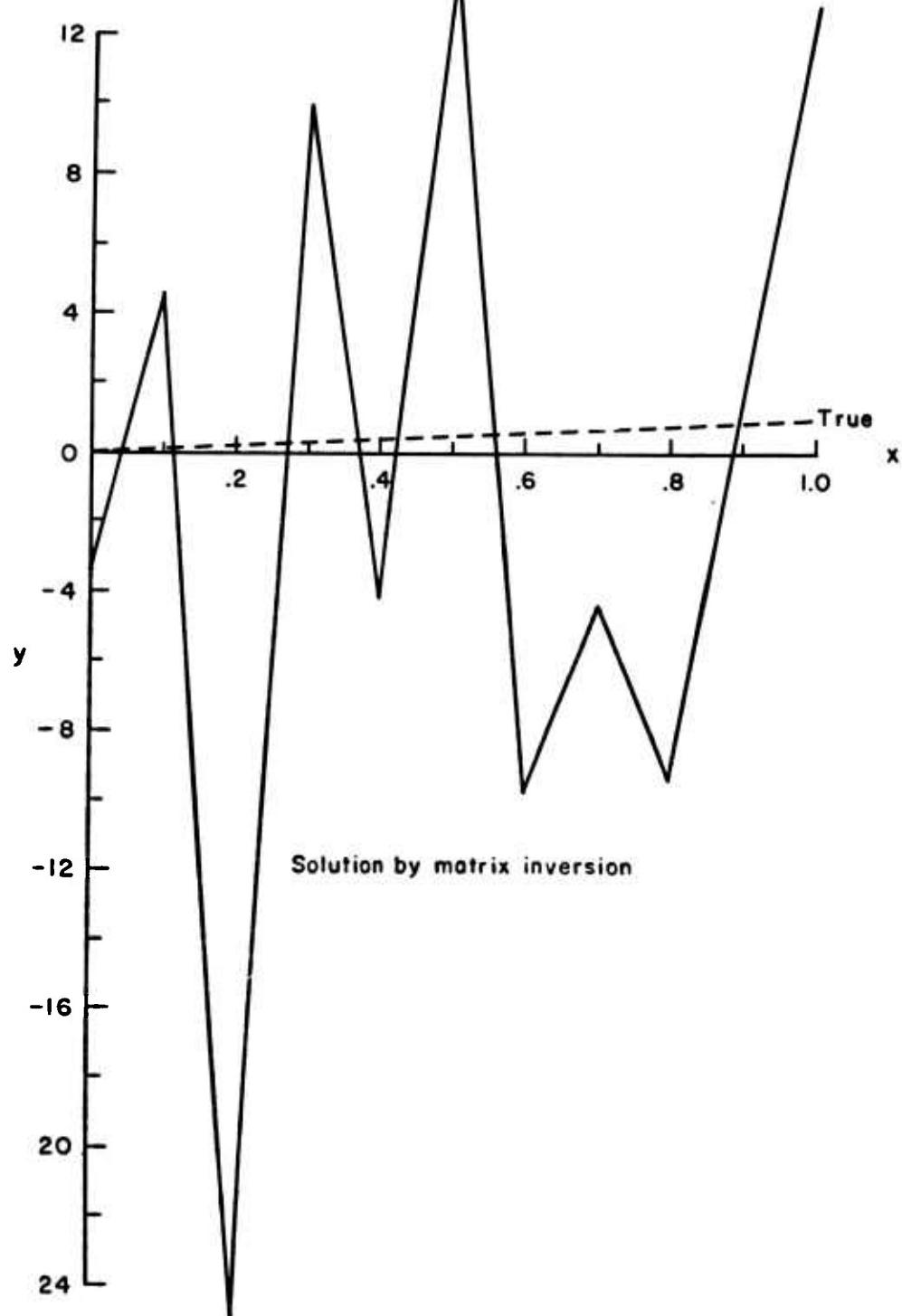


Fig. 1

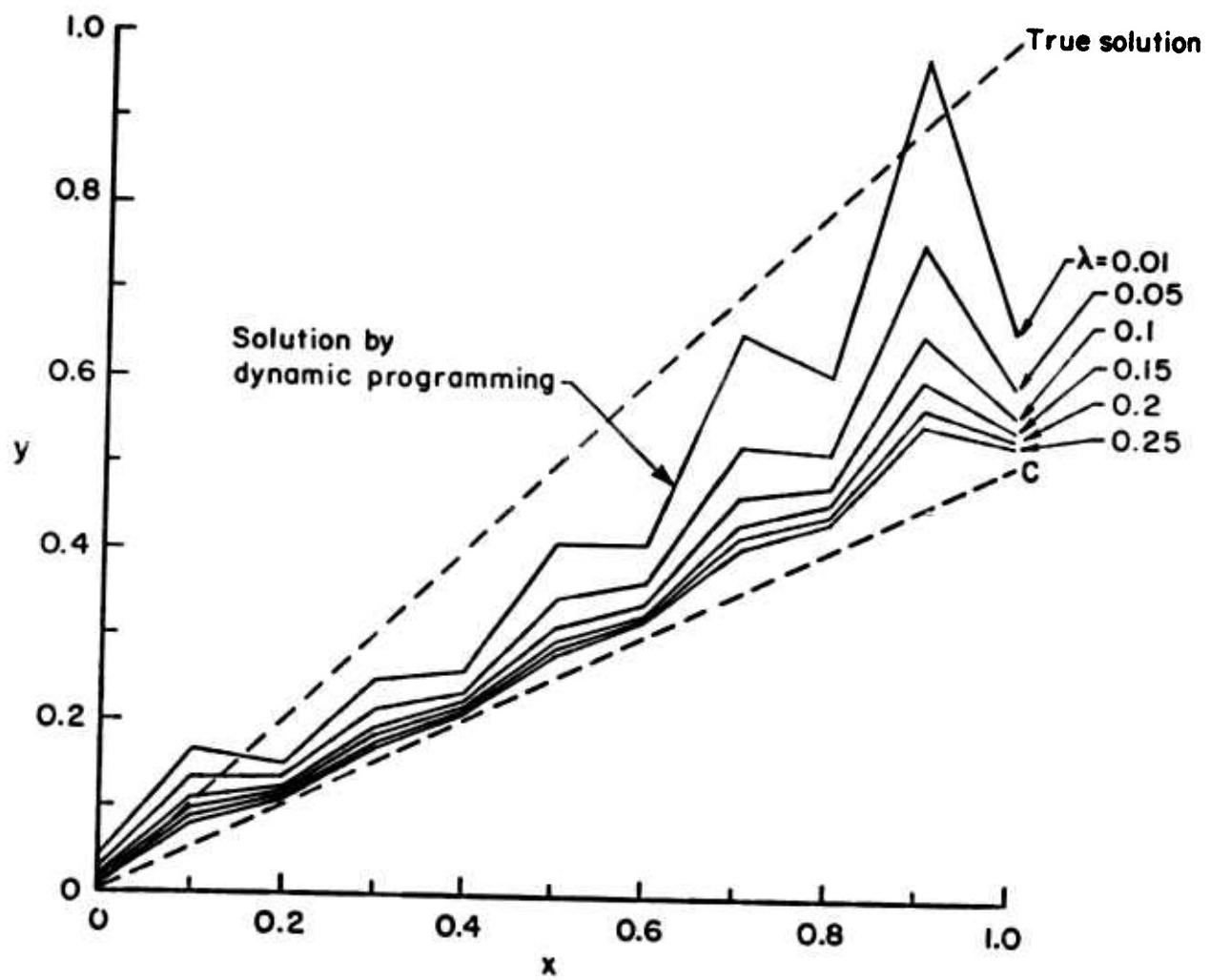


Fig. 2

From (4.1), we see that

$$(4.2) \quad (A'A + \lambda I)x_N = A'b + \lambda x_{N-1}.$$

Since $A'A + \lambda I$ is never singular for $\lambda > 0$, we have

$$(4.3) \quad x_N = (A'A + \lambda I)^{-1}A'b + \lambda(A'A + \lambda I)^{-1}x_{N-1}.$$

Since A is nonsingular, by assumption, the characteristic roots of $A'A + \lambda I$ are all greater than λ . The sequence in (4.3) therefore converges, geometrically, to a vector x . From (4.3),

$$(4.4) \quad (A'A + \lambda I)x = A'b + \lambda x,$$

whence, since A' is nonsingular, $Ax = b$.

In the application of (4.3), the choice of λ is crucial. If λ is too large, the convergence of (4.3) is so slow that computational error destroys accuracy; if λ is too small, the ill-conditioning of $A'A + \lambda I$ causes grave difficulty. We shall see how to overcome these difficulties to some extent, below.

5. APPLICATION OF SUCCESSIVE APPROXIMATION PLUS SMOOTHING

In Figure 3, we see the result of forty iterations starting with an initial approximation corresponding to $y/2$, with a value of $\lambda = 0.01$. Observe that the oscillations do not damp out in any strong fashion. If we continued the iteration, round-off error would soon eliminate any significance.

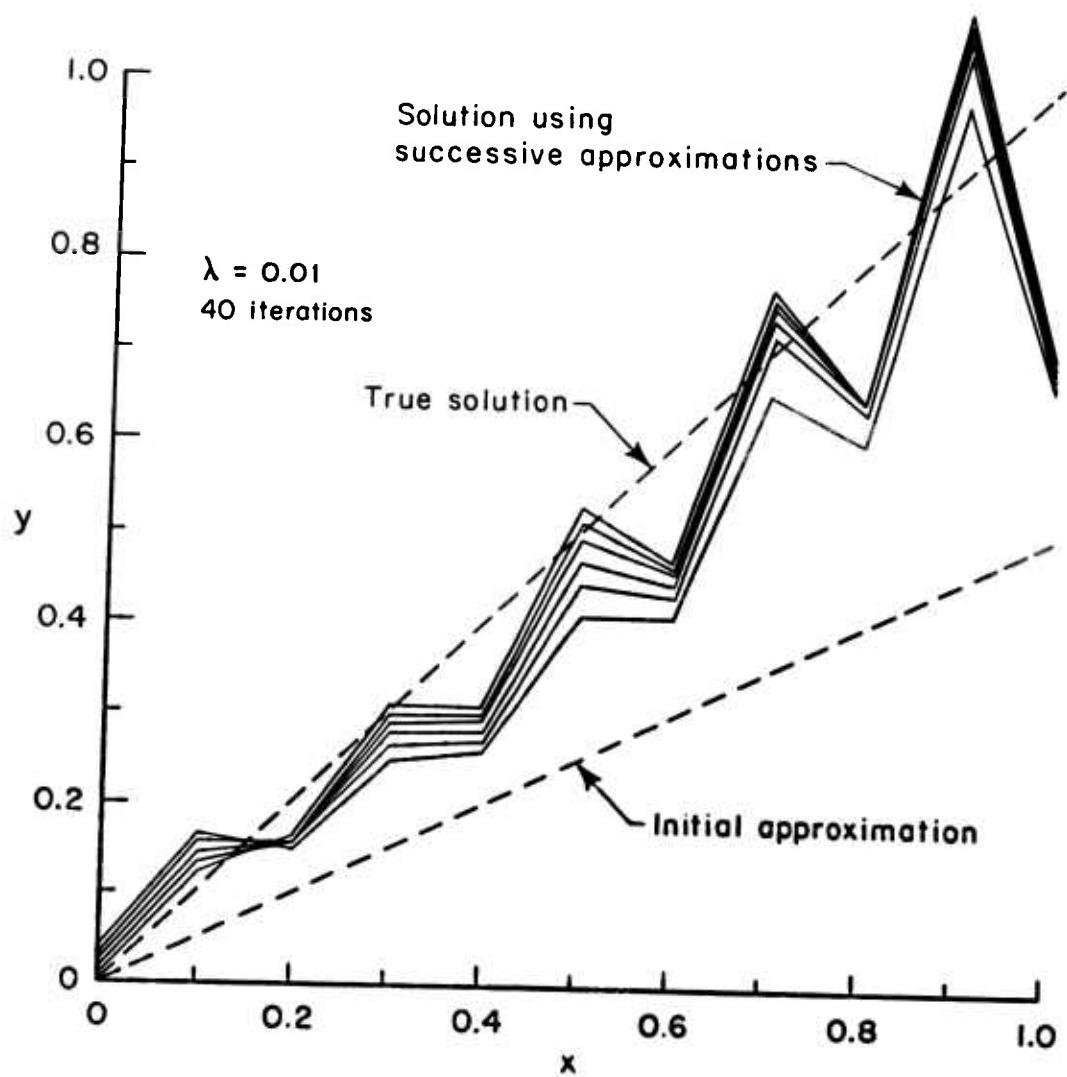


Fig. 3

Let us then smooth the solution and use this smoothed vector as a new initial approximation. No sophistication was used in the smoothing. Figure 4 shows the great improvement in accuracy combined with a diminution of oscillation.

This process could be repeated several times, and indeed made part of the original program. It is a simple example of the concept of sequential computation.

6. EXTRAPOLATION

Returning to the problem of minimizing

$$(6.1) \quad (\mathbf{Ax} - \mathbf{b}, \mathbf{Ax} - \mathbf{b}) + \lambda(\mathbf{x} - \mathbf{c}, \mathbf{x} - \mathbf{c})$$

for fixed \mathbf{c} , we know that the minimizing vector $\mathbf{x}(\lambda)$ approaches \mathbf{x}_0 , the solution of $\mathbf{Ax} = \mathbf{b}$. From the analytic expression for $\mathbf{x}(\lambda)$,

$$(6.2) \quad \mathbf{x}(\lambda) = (\mathbf{A}'\mathbf{A} + \lambda\mathbf{I})^{-1}(\mathbf{b} + \lambda\mathbf{c}),$$

we see that as $\lambda \rightarrow 0$,

$$(6.3) \quad \mathbf{x}(\lambda) = \mathbf{x}_0 + \mathbf{x}_1\lambda + \mathbf{x}_2\lambda^2 + \dots$$

Hence, from the values of $\mathbf{x}(\lambda)$ for small λ , we should be able to estimate \mathbf{x}_0 .

The point of the foregoing approach is that it is easier to compute $\mathbf{x}(\lambda)$ for $\lambda > 0$ because $\mathbf{A}'\mathbf{A} + \lambda\mathbf{I}$ is better conditioned than \mathbf{A} . The basic philosophy is to determine the solution in a convenient region and then

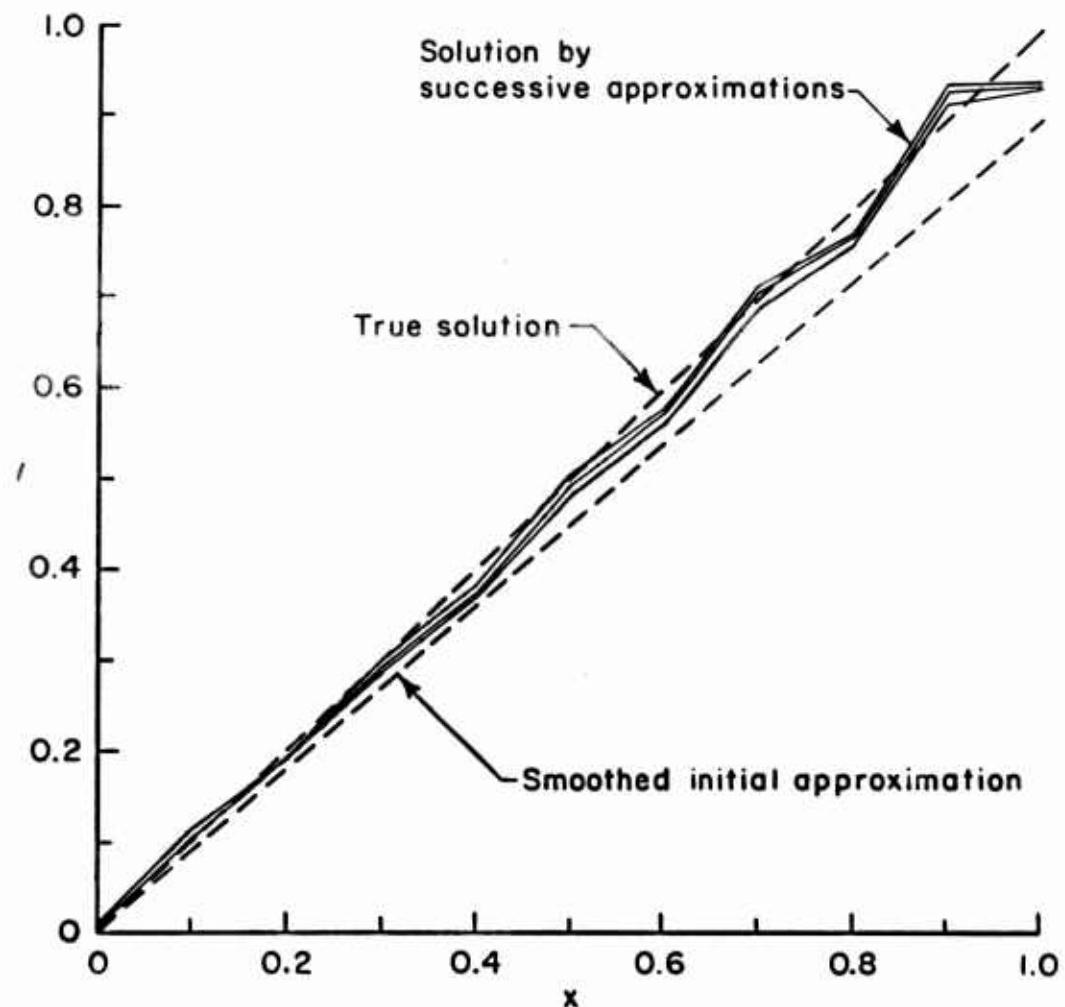


Fig. 4

use analytic continuation (here extrapolation) to obtain the solution in the desired region. For other applications of this method, see [5].

In Fig. 5 we show the dependence of $x(\lambda)$ for a range of values of λ , where $x(\lambda)$ has been obtained using the foregoing dynamic programming algorithm. In Fig. 6, we show the relation of the extrapolated solution, where again no sophistication was used in the estimation of x_0 , to the exact solution.

7. COMBINATION OF METHODS

Smoothing this approximate solution, we have an excellent first approximation for use in a successive approximation scheme. It is clear that we can combine these techniques in various ways.

8. DISCUSSION

It is generally agreed that no one technique will resolve the fundamental problem of obtaining sensible results from ill-conditioned systems. What we have wished to indicate in the foregoing pages is that control techniques with dynamic programming, successive approximations, extrapolation, and smoothing can yield worthwhile results in various cases.

There is no necessity to use dynamic programming algorithms for the determination of the minimum, and, in some cases, it may be more convenient to use standard

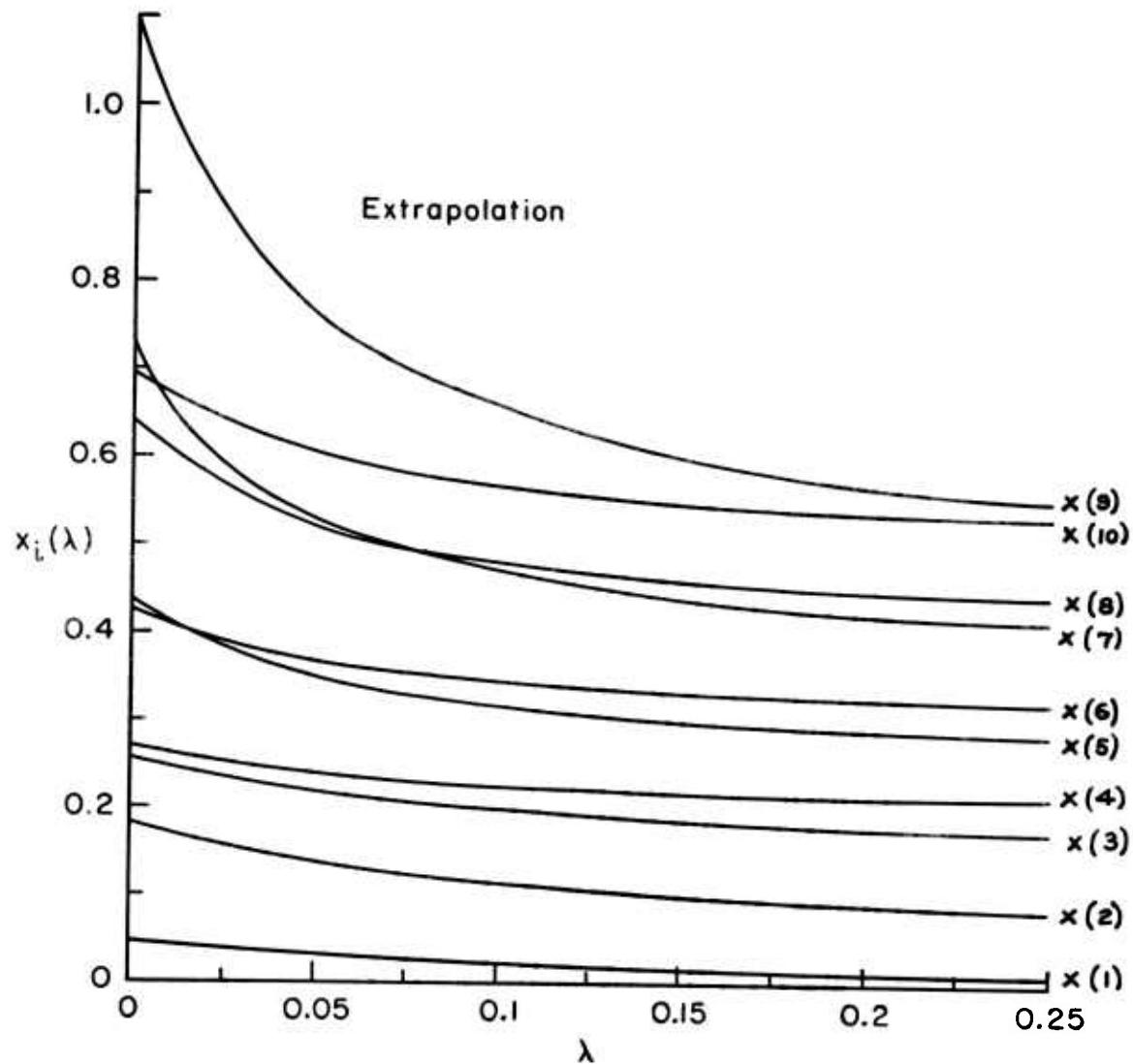


Fig. 5

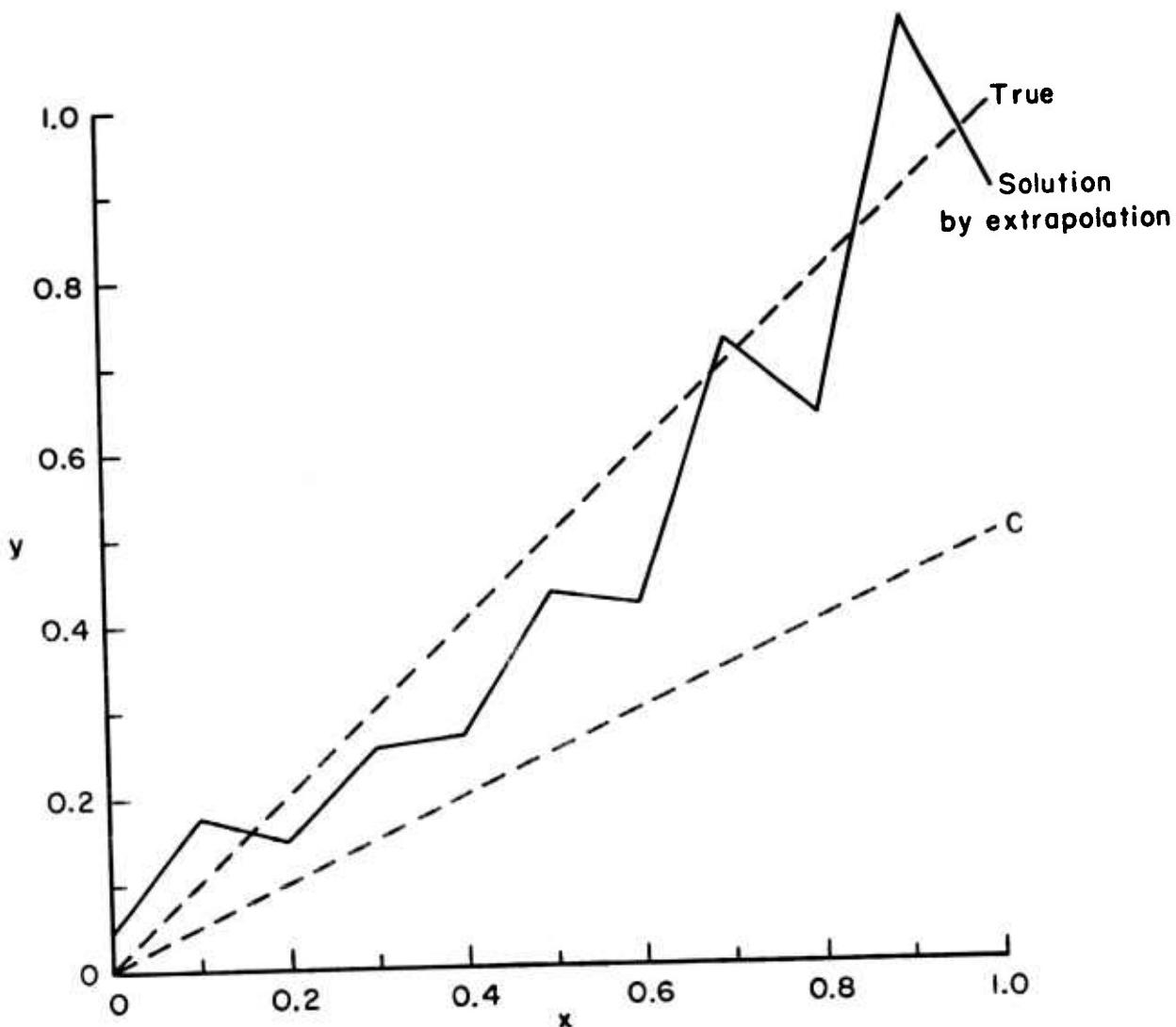


Fig. 6

algorithms. We would like to emphasize that a dynamic programming approach has a built-in stability, as indicated in the results of Sec. 3, and that it may be desirable for this reason to use it in many cases.

-19-

Appendix

COMPUTING ROUTINE

The computing routine used in this Memorandum is given
on the following pages.

```
C DYNAMIC PROGRAMMING SOLUTION
C MIN ((AX-B,AX-B)+K(X-C,X-C))
C QUADRATURE--SIMPSONS RULE
    DIMENSION C(11),B(11),XX(11),XY(11),W(11),A(11,11),XIDENT(11,11),
1   XK(11),XLAM(12),AKRN(11,11),Q(11,11),P(11),ALPHA(11,11)
2,RHO(11),X(11),G(11),XC(11)
1 FORMAT (3I12)
2 FORMAT (E12.8)
3 FORMAT (1H1,6GX,13HMATRIX A(I,J)///)
4 FORMAT (11E12.3)
5 FORMAT (1H0,40X,8HB VECTOR,34X,8HC VECTOR///)
6 FORMAT (39X,E10.3,32X,E10.3)
7 FORMAT (1H0,2UX,8HCASE NO. I5,7X,7HLAMBDA=E12.3///)
8 FORMAT (34X,2HX(I2,3H) =E14.5)
9 FORMAT (1H0,58X,17HMATRIX ALPHA(I,J)///)
10 FORMAT (1H0,39X,10HRHO VECTOR,33X,8HK VECTOR///)
C NUMBER OF LAMBDRAS (NLAM) AND MATRIX DIMENSION N
    READ 1, NLAM,N,ITER
    READ 2, (XLAM(I), I=1,NLAM)
C APPROXIMATE SOLUTION (C)
    READ 2, (C(I), I=1,N)
    DO 905 I=1,N
        XC(I)=C(I)
905 CONTINUE
C VECTOR B
    READ 2, (B(I), I=1,N)
    DO 900 I=1,N
        G(I)=B(I)
900 CONTINUE
C CALCULATION OF MATRIX A
    DO 100 I=1,N
        XI=I
        XX(I)=(.1*XI)-.1
        XY(I)=XX(I)
100 CONTINUE
    DELTA=.1.
    W(1)= DELTA/3.
    W(N)=DELTA/3.
    M=N-1
    DO 101 I=2,M,2
        W(I)=(4.*DELTA)/3.
101 CONTINUE
    M=N-2
    DO 102 I=3,M,2
        W(I)=(2.*DELTA)/3.
102 CONTINUE
    DO 103 I=1,N
        DO 103 J=1,N
            A(I,J)=((XX(I)-XY(J))**2)*W(J)
103 CONTINUE
    PRINT 3
    PRINT 4, ((A(I,J), J=1,N), I=1,N)
    PRINT 5
    PRINT 6, (B(I),C(I), I=1,N)
C DETERMINE Q(1), P(1), AND R(1)
C
```

```
C IDENTITY MATRIX
    DO 104 I=1,N
    DO 104 J=1,N
    XIDENT(I,J)=0.
104 CONTINUE
    DO 105 I=1,N
    XIDENT(I,I)=1.
105 CONTINUE
C
    DO 500 IXX5=1,NLAM
    DO 852 I=1,N
    C(I)=XC(I)
852 CONTINUE
    DO 850 IXX7=1,ITER
    DO 901 I=1,N
    B(I)=G(I)
901 CONTINUE
C KRONECKER CROSSPRODUCT
    XK(1)=0.
    DO 106 I=1,N
    XK(1)=XK(1)+(A(I,1))**2
106 CONTINUE
C
    DO 107 I=1,N
    DO 107 J=1,N
    AKRN(I,J)=(A(I,1)*A(J,1))/(XLAM(IXX5)+XK(1))
107 CONTINUE
C MATRIX Q(1)
    DO 108 I=1,N
    DO 108 J=1,N
    Q(I,J)=XIDENT(I,J)-AKRN(I,J)
108 CONTINUE
C
C VECTOR P(1)
    DO 109 I=1,N
    P(I)=(-XLAM(IXX5)*C(1)*A(I,1))/(XLAM(IXX5)+XK(1))
109 CONTINUE
C
C R(1)
    R=XLAM(IXX5)*(C(1)**2)-((XLAM(IXX5)*C(1))**2)/(XLAM(IXX5)+XK(1))
C
C
C DETERMINE ALPHA(1),RHO(1)
    DO 110 I=1,N
    ALPHA(I,1)=A(I,1)
110 CONTINUE
C
    RHO(1)=0.
C
C
C BEGIN ITERATION OF RECURRENCE RELATIONSHIPS
C DETERMINE ALPHA(STORE COLUMNWISE), RHO, AND XK
    DO 200 IXX1=2,N
    DO 201 I=1,N
    ALPHA(I,IXX1)=0.
    DO 202 J=1,N
    ALPHA(I,IXX1)=ALPHA(I,IXX1)+Q(I,J)*A(J,IXX1)
```

```
202 CONTINUE
201 CONTINUE
C
C
    RHO(IXX1)=0.
    DO 203 I=1,N
        RHO(IXX1)=RHO(IXX1)+P(I)*A(I,IXX1)
    203 CONTINUE
C
C
    XK(IXX1)=0.
    DO 204 I=1,N
        XK(IXX1)=XK(IXX1)+(ALPHA(I,IXX1)*A(I,IXX1))
    204 CONTINUE
C
C
    C DETERMINE KRONECKER CROSSPRODUCT OF ALPHA
    DO 205 I=1,N
        DO 205 J=1,N
            AKRN(I,J)=((ALPHA(I,IXX1)*ALPHA(J,IXX1))/(XLAM(IXX5)+XK(IXX1)))
    205 CONTINUE
C
C CALCULATE NEW Q
C
    DO 206 I=1,N
        DO 206 J=1,N
            Q(I,J)=Q(I,J)-AKRN(I,J)
    206 CONTINUE
C
C CALCULATE NEW P
C
    DO 207 I=1,N
        P(I)=P(I)- ((XLAM(IXX5)*C(IXX1)+RHO(IXX1)) *ALPHA(I,IXX1))/(
    1(XLAM(IXX5)+XK(IXX1)))
    207 CONTINUE
C
C CALCULATE NEW R
C
    R=R+XLAM(IXX5)*C(IXX1)**2-((XLAM(IXX5)*C(IXX1)+RHO(IXX1))**2)/
    1(XLAM(IXX5)+XK(IXX1))
200 CONTINUE
    PRINT 7, (IXX5,XLAM(IXX5))
    PRINT 9
    PRINT 4,((ALPHA(I,J), J=1,N), I=1,N)
    PRINT 10
    PRINT 6, (RHO(I),XK(I),I=1,N)
C
C CALCULATE X(N), X(N-1),...,X(1)
C
    DO 300 IXX2=1,N
        IXX3=N+1-IXX2
        X(IXX3)=(XLAM(IXX5)*C(IXX3)+RHO(IXX3))/(XLAM(IXX5)+XK(IXX3))
        DO 301 I=1,N
            X(IXX3)=X(IXX3)+(ALPHA(I,IXX3)*B(I))/(XLAM(IXX5)+XK(IXX3))
    301 CONTINUE
C
C CALCULATE NEW VALUES OF B
```

```
C
DO 302 I=1,N
B(I)=B(I)-X(IXX3)*A(I,IXX3)
302 CONTINUE
300 CONTINUE
PRINT 8, (I,X(I), I=1,N)
DO 851 I=1,N
C(I)=X(I)
851 CONTINUE
850 CONTINUE
500 CONTINUE
CALL EXIT
END
```

RS X060 - An F402 - Matrix Inversion with Accompanying
Solution of Linear Equations.

FAP 709/7090 Routine.

Source: Burton S. Garbow, Applied Mathematics Division,
Argonne National Laboratory, Lemont, Illinois, dated
February 23, 1959, distributed as 704 SHARE Routine
No. 664.

Revised by Susan Belcher, The RAND Corporation,
September 11, 1962.

REFERENCES

1. Twomey, S., "On the Numerical Solution of Fredholm Integral Equations of the First Kind by the Inversion of the Linear System Produced by Quadrature," J. Assoc. Computing Machinery, Vol. 10, 1963, pp. 97-101.
2. Phillips, D. L., "A Technique for the Numerical Solution of Certain Integrals Equations of the First Kind," J. Assoc. Computing Machinery, Vol. 9, 1962, pp. 84-97.
3. Bellman, R., An Introduction to Matrix Analysis, McGraw-Hill Book Company, Inc., New York, 1960.
4. Bellman, R., and S. Dreyfus, Applied Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1962.
5. Bellman, R., and R. Kalaba, "A Note on Nonlinear Summability Techniques in Invariant Imbedding," J. Math. Anal. Appl., Vol. 6, 1963, pp. 465-472.

0426455

0426455

GENERAL PURPOSE SUMMARY CARD

3